

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**  
**APPLICATION FOR LETTERS PATENT**

**INVENTOR:**

Cotner et al.

**TITLE:**

High Performance Support for XA Protocols in a Clustered Shared Database

## BACKGROUND OF THE INVENTION

### Field of Invention

The present invention relates generally to the field of relational databases. More specifically, the present invention is related to a shared memory device aiding in the  
5 implementation of robust 2-phase commit protocols.

### Discussion of Prior Art

The Open Group's XA protocol has become a computer industry standard for performing 2-phase commit operations between transaction managers and resource  
10 managers. Figure 1 illustrates a functional relationship between transaction manager (e.g., WebSphere®, WebLogic®, etc.) **102** and resource manager (e.g., DB2®, Oracle®, SQL Server, etc.) **104** on the Unix® and Microsoft Windows® platforms. Resource manager (RM) **104** is responsible for managing a part of a computer's shared resources (i.e., software entities can request access to a resource from time to time, using services  
15 that the RM provides), while transaction manager **102** is responsible for managing global transactions, coordinating the decision to commit them or roll them back, and coordinating failure recovery.

Transaction manager **102** and resource manager **104** use a 2-phase commit with  
20 presumed rollback. In a first phase, transaction manager **102** asks resource manager **104** to prepare to commit transaction branches (i.e., resource manager **104** is queried to see if

it can guarantee the ability to commit a transaction branch). If resource manager **104** is able to commit, it records any pertinent information it needs to do so, then replies affirmatively. A negative reply indicates failure of a transaction. After making a negative reply and rolling back its work, resource manager **104** can discard any  
5 knowledge it has of the transaction branch.

In a second phase, transaction manager **102** issues resource manager **104** an actual request to commit or roll back the transaction branch. Prior to issuing requests to commit, transaction manager **102** records decisions to commit, as well as a list of all  
10 involved resource managers (in this case, resource manager **104**). Resource manager **104** either commits or rolls back changes to resources and then returns status to the transaction manager **102**. Transaction manager **102** can then delete entries related to the global transaction.

15 Although XA is an industry standard, it is not nearly as robust as some of the proprietary 2-phase commit protocols that have been developed on other platforms such as OS/390 (e.g., IBM's systems network architecture (SNA) 2-phase commit used by Information Management Service (IMS) and IBM's customer information control system (CICS), resource recovery services (RRS) 2-phase commit used by WebSphere and DB2,  
20 and distribution relational database architecture (DRDA) 2-phase commit used by the DB2 family of products).

Provided below, and as depicted in figure 2, the following examples illustrate some of the scenarios where the XA protocol is less robust than some of the proprietary 2-phase commit protocols:

5

- XA requires that transaction manager **102** drive the XA RECOVER algorithm **206** to resolve indoubt units of work. Transaction manager **102** calls XA\_RECOVER() algorithm during recovery to obtain a list of transaction branches that are currently in a prepared or heuristically completed state. It should be noted that there is no provision for resource manager **104** to initiate the resolution of an indoubt unit of work.  
10
- XA RECOVER **206** requires that resource manager **104** provide a full list of indoubt transactions, but it has no provision where the members of a database server cluster can resolve indoubt units of work individually with the transaction manager **102**. Figure 3 specifically illustrates this scenario, wherein a full list of indoubt transactions **302** are passed on to XA RECOVER algorithm **303**, but a member **304** of a database server cluster is unable to resolve individual indoubt units of work **306**, **308**, and **310** with transaction manager **102**.  
15
- Indoubt units of work are typically resolved in XA during transaction manager **102** restart, as there's very little support for automatically  
20

resolving an indoubt unit of work that occurs due to a communication failure on a single network connection (i.e., only 1 of the "n" communication connections failed), without restarting the transaction manager **102**.

5

Furthermore, database systems are increasingly using hardware clustering technology to improve the overall availability of database servers. When database systems exploit clustering, they strive to provide a single-system image for the cluster of server machines, so that the clients (such as an XA transaction manager) are unaware that  
10 multiple physical machines are being used to run the database product. This creates a dilemma for satisfying the above-mentioned XA RECOVER requirement (i.e., that any member of the database cluster must be able to provide a full list of indoubt transactions upon demand, and this list must include indoubt transactions for all members of the database server cluster) while still allowing the XA RECOVER to occur when one or  
15 more members of the database server cluster are not available. Listed below are a few techniques currently available to address this requirement, but it should be stressed that each of these techniques have their own limitations:

a) client-side logging -- this approach has the database client middleware write special log records on the client that record the list of indoubt transactions,  
20 wherein the database client middleware is able to consult the log to obtain a full list of indoubt transactions, without relying upon the availability of any of

the database server members (it should be noted that one disadvantage of this approach is that the database client log becomes an object that must be handled for application server failover planning, backup, recovery, etc. and, moreover, this approach introduces a lot of additional administrative overhead for the customer).

b) server-side indoubt table -- with this approach, the database client middleware performs INSERTs and DELETEs with a special table at the database server to keep track of indoubt transactions across the members of the database cluster (it should be noted that although this approach solves the above-mentioned administration overhead issues, it has negative performance implications in that additional INSERT and DELETE operations have to be performed to the relational table, with such operations introducing additional logging, etc.).

c) database cluster support for XA RECOVER -- the database engine can provide support for the merged list of indoubt transactions through various means:

- a single log stream that contains the log records from all the members of the database cluster;
- a special table containing the indoubt units of work at any given point in time; and

- special XA RECOVER logic that merges the logs produced by all the members of the database cluster to produce a unified list of indoubt transactions for the cluster.

5        Each of the above-mentioned techniques would have negative performance or scalability implications for the database cluster. Hence, there exists a need to resolve the XA RECOVER requirement in a cluster of database servers:

- without requiring all the members of the database cluster to be active during XA RECOVER -- without introducing significant added CPU or  
10        elapsed time cost for processing the database transactions; and
- without limiting the scalability of the database cluster.

The following references provide for a general teaching in the area of distributed computing and database configuration.

15

The U.S. patent to Slaughter et al. (6,014,669), assigned to Sun Microsystems, provides for a highly-available distributed cluster configuration database. The cluster configuration database is a distributed configuration database wherein a consistent copy of the configuration database is maintained on each active node of the cluster. Each node  
20    in the cluster maintains its own copy of the configuration database and configuration database operations can be performed from any node. Configuration database updates

are automatically propagated to each node in a lock-step manner. If any node experiences a failure, the configuration database uses a reconfiguration protocol to insure consistent data in each node of the cluster.

5       The U.S. patent to Badovintz et al. (5,805,786), assigned to International Business Machines, provides for the recovery of a name server managing membership of a domain of processors in a distributed computer environment which includes detecting the failure of the name server node and consulting a membership list of nodes in the domain to determine the crown prince (CP) node who is next in line to become the name  
10   server. The other available nodes in the domain periodically send recover messages to the CP node, and responsive to receiving the recover messages from all the other available nodes in the domain, the CP node perform a two phase takeover whereby the CP node becomes the name server for managing said processors in the domain. After the CP node becomes the name server, the other available nodes in the domain send data to the new  
15   name server necessary for the name server to manage the other available nodes in the domain. All request messages requesting management by the name server are stored locally until after the CP becomes the name server. The locally stored request messages are then processed by the other available nodes such that no request messages are lost during recovery. U.S. patents 5,896,503 and 5,790,788, also assigned to International  
20   Business Machines, provide for similar teachings.



The patent to Attanasio et al. (5,668,943), assigned to International Business Machines, provides for a system and method for recovering from failures in the disk access path of a clustered computing system. Each node of the clustered computing system is provided with proxy software for handling physical disk access requests from applications executing on the node and for directing the disk access requests to an appropriate server to which the disk is physically attached. The proxy software on each node maintains state information for all pending requests originating from that node. In response to detection of a failure along the disk access path, the proxy software on all of the nodes directs all further requests for disk access to a secondary node physically attached to the same disk.

The patent publication to Jacobs et al. (2003/0018732) discloses a method for replicating data over a network using a one or two phase method. For the one phase method, a master server containing an original copy of the data sends a version number for the current state of the data to each slave on the network so that each slave can request a delta from the master. The delta that is requested contains the data necessary to update the slave to the appropriate version of the data. For the two phase method, the master server sends a packet of information to each slave. The packet of information can be committed by the slaves if each slave is able to process the commit. Patent publication 2003/0023898, also by Jacobs et al., provides for a similar teaching.

The Japanese patent to Brockmeyer et al., assigned to International Business Machines, discloses an expansion function of the two-phase commit protocol which attains the subscription of distributed subscribers between physically separated agents without relying upon the communication mechanism used in data processing systems.

5

The non-patent literature to Svobodova entitled, "File Servers for Network-Based Distributed Systems," discloses a file server that provides remote centralized storage with options for performing an atomic update of data stored in the file server.

10

The non-patent literature to Mohan et al., entitled "Method for Distributed Transaction Commit and Recovery Using Byzantine Agreement Within Clusters of Processors," replaces the second phase of one of the commit algorithms with a Byzantine agreement, allowing for certain trade-offs and advantages at the time of commit (thereby providing speed advantages at the time of recovery from failure).

15

The non-patent literature to Wang et al. entitled, "A Mobile Agent Based Protocol for Distributed Database Access," provides for a three-tier protocol to improve data transmission while accessing distributed databases.

The non-patent literature to Hsial entitled, "DLFM: A Transactional Resource Manager," provides for a two-phase commit protocol and a scheme for enabling rolling back a transaction update after a commit to the local database.

5 Chapter 14 of the book entitled "Advanced Database Systems" provides a review of parallel recovery in replicated databases.

Whatever the precise merits, features, and advantages of the above cited references, none of them achieve or fulfills the purposes of the present invention.

10

#### SUMMARY OF THE INVENTION

The present invention provides for a shared memory device called the Coupling Facility (CF) used to record the indoubt transaction entries for each member of the database cluster, avoiding the CPU cost and elapsed time impact of persisting this  
15 information to disk (either via a log write or a relational table I/O). The CF provides full read/write access and data coherency for concurrent access by all the members in the database cluster. At any given point in time, the CF will contain the full list of indoubt transactions for the entire database cluster. The approach of the present invention has numerous advantages over the prior art, some of which include: no additional logging  
20 overhead, no additional SQL statements, very minimal CPU cost or I/O latency due to the store/delete operations against the CF for the indoubt list, the CF can provide the full list

of indoubt transactions even when one or more members of the database cluster are unavailable, and excellent scalability characteristics, since each member of the database cluster can operate with independent log streams and minimal contention on the CF structure.

5

In an extended embodiment, CF duplexing is used to guarantee the integrity of the CF structure used for the indoubt list. In the event of complete loss of both CF structures (which will not happen except in major disaster situations), data sharing group restart processing can reconstruct the CF structures from the individual member logs.

10

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a functional relationship between a transaction manager and a resource manager.

Figure 2 illustrate a sample scenario where the XA protocol is less robust than  
5 other proprietary 2-phase commit protocols.

Figure 3 illustrates a drawback of the XA protocol in its inability to address and resolve each individual indoubt transaction.

Figure 4a illustrates another embodiment of the system of the present invention.

Figure 4b illustrates a method associated with the embodiment of figure 4a.

10

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

While this invention is illustrated and described in a preferred embodiment, the invention may be produced in many different configurations. There is depicted in the drawings, and will herein be described in detail, a preferred embodiment of the invention,  
15 with the understanding that the present disclosure is to be considered as an exemplification of the principles of the invention and the associated functional specifications for its construction and is not intended to limit the invention to the embodiment illustrated. Those skilled in the art will envision many other possible variations within the scope of the present invention.

20

The present invention provides for a system and method Figure 4a illustrates a system based upon the present invention that supports the XA 2-phase commit protocol in a clustered shared database. In this embodiment, a special shared memory device called the Coupling Facility (CF) **407** is used to record the indoubt transaction entries, in  
5 a coupling facility list structure **408**, for each member of the database cluster **402**, **404**, and **406**. Having the CF avoids the CPU cost and elapsed time impact of persisting this information to disk (either via a log write or a relational table I/O). The CF provides full read/write access and data coherency for concurrent access by all the members in the database cluster (represented in figure 4 as **402**, **404**, and **406**). At any given point in  
10 time, the CF will contain the full list of indoubt transactions for the entire database cluster. Transaction manager **412** communicates with a client (e.g., a DB2 client), which in turn uses network **410** to communicate with a database (e.g., DB2) in the database cluster **402**, **404**, and **406**. The database is then able to manipulate the CF **407** as needed.

15 Figure 4b illustrates method **414** associated with the system of Figure 4a. During first time startup/install of the database system, in step **416**, the coupling facility (CF) creates a coupling facility list structure for recording the XA indoubt list entries. In one embodiment, CF duplexing is used to insure that the list structure survives almost all feasible error scenarios.

20

In step **418**, an XA transaction receives a "PREPARE TO COMMIT" message from the XA transaction manager on a database transaction that is not read-only. In step **420**, the database system adds an entry in the CF list structure containing, in one embodiment, the XA global transaction identifier (GTRID), the timestamp when the entry was created, and a state indicating the transaction is INDOUBT. After creating this entry in the CF list structure, the database will record the normal log record indicating the unit of work is indoubt. Next, in step **422**, the CF receives an invocation for a second phase of commit (commit or rollback decision) from the XA transaction manager.

10       After the completion of step **420**, one of three steps follows. In step **424**, the database server receives a COMMIT decision from the XA transaction manager. When this occurs, in step **426**, the database engine records the commit decision on the log and, in step **428**, deletes the entry in the CF list structure.

15       In step **430**, a ROLLBACK decision is received from the XA transaction manager. When this occurs, in step **432**, the database engine records the rollback decision on the log and, in step **434**, deletes the entry in the CF list structure.

The network connection to the XA transaction manager or the database system  
20   can terminate abnormally before completing steps **424-428** or **430-434**. If this occurs, as

shown in steps **436-440**, the CF list structure is used to respond to requests from the XA transaction manager to perform the XA RECOVER command.

If the network fails, the database system simply waits for the XA transaction  
5 manager to reconnect and perform the XA RECOVER command. If the database system terminates abnormally, prior to deleting the entry in the CF list structure, the database system will be restarted and go through both forward recovery and backward recovery.

Three possible situations can occur. If a commit or rollback decision is found on  
10 the log and the CF list entry still exists, the database system crashed after writing the log record (but before deleting the CF list entry). In this case, the database system simply deletes the CF list structure entry and continues restart processing.

If no commit or rollback decision is found on the log, the database examines the  
15 CF list structure entry to see if a commit or rollback decision has been received by another member of the database server cluster. If a commit or rollback decision was received, the database engine records the appropriate log record and deletes the CF structure entry.



If no commit or rollback decision is found on the log and no commit/rollback decision is present in the CF structure entry, the unit of work is still indoubt, so the database waits for the XA transaction manager to perform the XA RECOVER command.

5        When the database system receives an XA RECOVER request, the database examines the CF list structure and reports the GTRID of each indoubt unit of work to the XA transaction manager.

      Upon receiving the list of indoubt GTRIDs, the XA transaction manager sends  
10    commit or rollback decisions for the units of work that are owned by the XA transaction manager. These decisions result in one of two outcomes. If the database cluster member that receives the decision actually owns the indoubt unit of work, the database will write the appropriate log record, delete the CF list structure entry, and complete the commit or rollback operation for the indoubt unit of work.

15

      If the database cluster member that receives the decision does not own the indoubt unit of work, the database records the commit or rollback decision in the CF entry and record a log record reflecting this activity. The database then sends a CF notify message to cause the database member that owns the unit of work to resolve the indoubt unit of  
20    work (if the member is currently available).

Additionally, the present invention provides for an article of manufacture comprising computer readable program code contained within implementing one or more modules providing high performance support for XA protocols in a clustered shared database. Furthermore, the present invention includes a computer program code-based product, which is a storage medium having program code stored therein which can be used to instruct a computer to perform any of the methods associated with the present invention. The computer storage medium includes any of, but is not limited to, the following: CD-ROM, DVD, magnetic tape, optical disc, hard drive, floppy disk, ferroelectric memory, flash memory, ferromagnetic memory, optical storage, charge coupled devices, magnetic or optical cards, smart cards, EEPROM, EPROM, RAM, ROM, DRAM, SRAM, SDRAM, or any other appropriate static or dynamic memory or data storage devices.

Implemented in computer program code based products are software modules for:

(a) creating a coupling facility list structure in a coupling facility; (b) receiving an instruction from a member in a database cluster to create an entry in the coupling facility list structure, wherein the instruction is sent by the member in the database cluster after receiving a PREPARE TO COMMIT message regarding a transaction from a transaction manager; (c) creating an entry corresponding to the transaction in the coupling facility list structure, wherein the entry comprises a global transaction identifier, a timestamp identifying when said entry was created, and a state indicating that said transaction is an

indoubt transaction; (d) deleting said entry corresponding to said transaction upon successful completion of either a COMMIT or ROLLBACK decision in said database cluster;

if, in step (d), communication link between said coupling facility and database cluster or if communication link between the coupling facility and transaction manager failed, then said coupling facility:

reestablishing communication links,  
receiving a RECOVER message from said transaction manager,  
transmitting a list of the global transaction identifiers to the transaction manager, wherein the transaction manager identifies associated indoubt entries, issues a COMMIT or ROLLBACK decision for said associated indoubt entries, and notifies other transaction managers corresponding to remainder of indoubt entries; and

wherein, upon successful execution of said issued COMMIT or ROLLBACK decision, the coupling facility receives an instruction from a member in said database cluster for deleting corresponding entry in coupling facility list structure.

## CONCLUSION

A system and method has been shown in the above embodiments for the effective implementation providing high performance support for XA protocols in a clustered shared database. While various preferred embodiments have been shown and described,  
5 it will be understood that there is no intent to limit the invention by such disclosure, but rather, it is intended to cover all modifications falling within the spirit and scope of the invention, as defined in the appended claims. For example, the present invention should not be limited by software/program, computing environment, or specific computing hardware.

10 The above enhancements are implemented in various computing environments. For example, the present invention may be implemented on a conventional IBM PC or equivalent, multi-nodal system (e.g., LAN) or networking system (e.g., Internet, WWW, wireless web). All programming and data related thereto are stored in computer memory, static or dynamic, and may be retrieved by the user in any of: conventional computer  
15 storage, display (i.e., CRT) and/or hardcopy (i.e., printed) formats. The programming of the present invention may be implemented by one of skill in the art of graphics or object-oriented programming.